INSTITUTO TECNOLÓGICO Y DE ESTUDIOS

SUPERIORES DE MONTERREY

Campus Monterrey

Departamento de Computación y Mecatrónica

Dibujo con letras blancas

Descripción generada automáticamente

**Design and testing of an SDN-based mitigation system to defeat steady DDoS attacks.**

*by*

Abraham Lemus Ruiz

Business Solution Development Capstone Project

Computer Systems

Engineering

Advisers:

Dr. Jesús Arturo Pérez Díaz

MC. Noé Marcelo Yungaicela Nava

Santiago de Querétaro, Querétaro, México

02/06/2021

Acknowledgement

Example:

Give thanks to advisors First of all I would like to give a special thanks to the project advisors. Especially to Dr. Jesus Arturo Perez Diaz for motivating me to work on a project related to computer security in spite of not being the area I have developed more in my university career. The results of this project are a reflection of the continuous support that was given week after week by the advisors. I also want to thank my mom and my brother for accompanying and supporting me in my last semester while I was doing this research work.

1. Abstract

The world of today is extremely connected, we have seen the rise of powerful technologies that were designed to meet the requirements of connectivity. To meet this demand, companies have adopted technologies such as the SDN (Software Defined Networks). SDN is a networking paradigm that abstracts the data plane from the control plane where a single controller device can manage the whole network. Despite its initial prospects, an SDN also has security vulnerabilities. A recent security measure that has been tried is to implement “new generation firewalls” that use ML techniques to identify and mitigate security attacks. This work will design and implement a new mitigation strategy for DDoS attacks in an SDN using a simulated environment and perform an evaluation of the said strategy according to its performance, response time, and scalability.

Contents

Introduction

**DDoS**

A Distributed Denial of Service (DDoS) attack targets an internet service and floods the network with many requests, when the victim cannot handle the traffic, its resources will be depleted, and legitimate users will not be able to use it. [1]

DDoS attacks can be divided into the following categories:

Application layer attacks: they exploit service vulnerabilities in the 7th layer and exhaust the resources of the target. After an attack of this type is executed, the server cannot accept new requests. “Slowloris” is a common example of this type of attack. [1]

Resource Exhaustion attacks: they take advantage of weaknesses in the network layer and consume the server’s resources such as the CPU and the memory. “TCP-SYN flood attack” is an instance of this type of attack. [1]

Volumetric attacks: they saturate the access points of a system, consuming all bandwidth of the target. “ICMP flood attack” is the most common type of volumetric attack. [1]

A business that has gone through a DDoS attack will suffer disastrous consequences, far beyond the loss of expected income for the problems of operational time. According to Cox Blue [2], on average, a DDoS attack costs between \$20k and \$40k USD per hour, however, the result may be the loss of confidence from its customers. The industry sector of a business is not a factor for attackers to decide whether to conduct an attack, anyone that depends on internet communication to conduct their activities is a potential target [2].

//Imagen de DDoS

**Slow Rate DDoS**

Slow rate attacks belong to the application layer category, and they work by exploiting the HTTP method. They are executed by sending many requests slowly to an online server and creating a bottleneck by not completing them. These requests will usually keep a connection to the server open for prolonged periods of times, thus depriving legitimate users of access to the server [12].

These attacks are usually done with the help of a script that continually sends these requests, they typically do not need a lot of bandwidth, which makes them potentially more dangerous, as the resource limitation is less compared to other kinds of attacks. Also they are relatively harder to detect because the attackers are behaving according to protocols and their behavior could be explained by poor connectivity on their end.

There are three main types of slow-rate DDoS attacks:

Slow Headers (Slowloris): They send HTTP headers slowly in an incomplete way, so the connection is kept open and further consume the server’s resources.

Slow Body: They are very similar to the slow headers attack, they work by sending a POST request very slowly to the server, also keeping alive their connections to make normal traffic impossible.

Slow Read: They work differently from the previous ones, where in this case, legitimate TCP-SYN traffic is sent to the victim requesting a resource from it, but they read the response very slowly, taking more network resources as time goes on.

**SDN**

More people are using the internet than ever before. According to Cisco, the number of devices connected to networks will be three times the global population by 2023 [3]. The difficulty in meeting the connectivity demands of the present and the future only increases when we consider that many of these devices are phones, TVs, smartwatches, etc., all with different connectivity requirements and processing power [4]. All these factors make it attractive for malicious entities to use DDoS attacks for different reasons.

Software-Defined Networks (SDN) are a technology that has been gaining popularity among organizations and researchers [5]. An SDN is characterized by being a flexible and dynamic architecture. Its distinctive quality is that it separates the control plane from the data plane, centralizing all network functions to a single controller unit [5].

An SDN can use the following strategies to mitigate security threats:

Flow Filtering: this technique looks up different fields from a flow entry such as source, destination addresses, and ports and straight out blocks it if it is deemed malicious by its IDS. The mitigation step depends on the statistics and inspection tasks done by the controller. [1]

Honeypots: it involves an isolated and insecure environment that looks like a real target, to lure potential attackers to it while keeping the real target safe. [1]

Rate Limiting: as the name suggests, this technique involves setting a threshold on the volume traffic. If this threshold is reached, the remaining network requests will be dropped. [1]

Moving Target Defense: with this strategy, some network parameters, such as IP addresses, are updated randomly over time to make it more difficult to locate the network's hosts and services. [1]

Traceback: when using traceback in an SDN, an attacker's origin can be found by looking at the packet's headers. [1]

Request Prioritization: with this technique, each source host is assigned a reliability factor depending on its history on the network. If a device has made many requests over a short time, it will be tagged as suspicious, and its reliability factor will decrease. The system administrator can choose to reject a packet if it is unreliable by a certain threshold. [1]

//Diagrama SDN

State of the Art

In [5], Tuan et al. worked with an SDN-based ISP network architecture to develop an ML-based mitigation strategy. They propose a system in which the controller calculates the entropy of each IP source. Then, a KNN model is used to define a threshold to separate the malicious traffic from the benign traffic. The network architecture used in the project consists of an SDN controller, an SDN enabled switch as well as some hosts connected. When the attacking flow is identified, the controller sends a packet to the corresponding switch with the instruction to block it. This approach was able to mitigate over 98\% of attack traffic; however, in the event of a TCP-SYN flood attack, each new flow must be processed and that might lead to overloading, considering that the detection and mitigation phases occur near instantly. This strategy is useful for TCP and ICMP flood attacks but may not be effective against Slow-rate DDoS attacks.

Another technique is the one designed and implemented by Assis et al. [4] where they implement a system with two modules with the tasks for detection and classification. In the detection module, the authors use a Gated Recurrent Unit (GRU) deep learning model to identify DDoS attacks individually, instead of the more common flow analysis; the traffic is classified into normal or malicious. If it gets the latter label, it is then passed to the mitigation module, which is divided into two submodules, one that identifies the optimal countermeasure to use (a drop flow rule), and the second submodule sends the optimal drop policy to the controller and is implemented as soon as it is received. The flow rule is generated by looking at different flow attributes such as IP addresses and ports. In this phase, IP flows are also processed individually, instead of processing them through broad time windows. This method decreases the overall mitigation time because it can just identify the problem node on the spot and discard all network flow related to it. The impact of mitigation is kept low because the mitigation step is done right after the individual IP attack detection step, and in general, the cost of mitigation is directly tied to the cost of detection and there are no complex probabilistic methods used. The results of Assis work show that the GRU is very efficient at detecting DDoS attacks, with an accuracy of over 95\%, comparable to other research in the area. For mitigation, Assis analyzes the legitimate flows dropped and the malicious flows not dropped, and in this case the results are not as good as the previous ones because of the amount of malicious traffic that stays on the network, but still decent when considering the sum of both malicious and non-malicious. Another disadvantage of this approach is the vast amount of data the system needs to analyze.

Novaes et al. [6] propose a similar system to [5], they implement an LSTM (long short-term memory) mechanism to detect anomalous traffic with an accuracy of 98\%. Their system is comprised of three phases: traffic characterization, which uses an LSTM model to determine “normality” among different traffic parameters such as the entropy for sources and destinations of IPs and ports, etc. The outcome of this phase is used for the second phase, anomaly detection, which uses fuzzy logic and applies a membership function to the analyzed attributes and compares to thresholds to classify traffic into three categories: normal, port-scan and DDoS. The third phase is the mitigation phase, when the IDS system detects an attack, the controller will use a model called Event Condition Action (ECA), where the event is an anomaly associated with a set of rules, the rules use different conditions related to said anomaly and an action is taken as a mitigation policy. The variables taken into consideration are the IP and port source and destination addresses as well as the protocols. The algorithm used for mitigation goes as follows: constantly monitor for attributes of legitimate packets and include them in a safe list, identify the most visited IP address among the anomalous flow and drop all packets directed to it if the incoming flow’s attributes are not in the safe list. The benefits of this approach are that it managed to get a 99\% drop rate of anomalous packages and the system always returned to normalcy. The drawback to this approach is that the model is not very adaptable, while the way attacks work is constantly updated, and it does not consider slow HTTP attacks such as slow-headers, slow-read or slow-loris, which are the focus of this work, however Novaes also showcases the use of multiple techniques such as a safelist, monitoring normal traffic, and the flow table modification, which helps in the work for different mitigation policies.

A strategy that mitigates slow HTTP attacks is the one proposed by Hong et al. [7] They implement a Slow HTTP DDoS attack Defense Application (SDHA) on an SDN controller. When a webserver detects a request has more open connections than an established threshold, the suspicious flows are sent to the SDHA via a flow rule update to perform detection using a timeout-based algorithm, where basically the defense application checks if the incomplete requests are not completed within certain amount of time. If the flow is deemed malicious, a flow rule update is issued to terminate all open connections from the source and to block future requests from the attacker. This method allowed the web server to have 180 maximum connections open from an attacker, thus it will not fall into a DoS state. The disadvantage of this approach is that the system is not autonomous, as the target hosts are required to start the attack check routine; another problem is that the thresholds and other parameters are static, so an attacker may vary the details of an attack to evade this defense.

In [8], Perez-Díaz et al. created an architecture for an SDN that detects and mitigates LR-DDoS attacks such as slow body, slow read, slow headers, Goldeneye, etc. The system is divided into two modules: an IPS that deals with flow and mitigation management and an IDS, that runs different ML and DL models to classify traffic into benign or malicious. If an attacker is detected, the IDS sends a json message to the IPS with the attack data and it is added to a blacklist with a drop probability of 10%, each successive detection will increase the probability by 5%, and when this value reaches 100, a drop rule is issued by the controller. The results showed that algorithms such as Random Forest, SVM reached accuracies of over 90% and MLP detecting up to 95% with less than 1% false positives; the mitigation module was able to drop attacks effectively where the drop probability was over 100\%. The advantages of Perez-Díaz work are that it is a complete IPS + IDS system, and that it was able to mitigate most of the detected attacks.

In [9], Pascoal et al. propose the slow saturation attack, a variant of the Slow TCAM attack (also proposed by him previously) that consumes a switch’s TCAM memory by forcing the installation of flow rules and preventing it from creating new rules for legitimate traffic while using techniques of a saturation attack, such as sending 100 packets per second and forcing the timeout of existing flow rules. The danger of this kind of attacks is that the attacker can use different tools (such as SDN probing methods) to know the SDN’s timeout configuration or other parameters to plan and execute the attack, thus circumventing basic detection and mitigation strategies. The authors propose several possible countermeasures, the most promising one is SIFT (Selective Defense for TCAM), which was built on top of the OpenFlow protocol. With this system, when the switches are getting overloaded, a probabilistic function is calculated to determine whether to install or to reject a new flow rule. If it is installed, the switch will drop a random existent flow rule as well as all its related flow rules. If the flow rule is not installed, the system will continue as normal. The idea behind this algorithm is that when there is an overload, the probability to randomly select an attacker flow increases. This system reached a rate of between 95% and 98% of clients recovering their connection over different flow rates. The problem with this approach for this work is that it deals with slow TCAM instead of slow rate, but there may be some compatibility in the strategies proposed; other problem is that it does not deal with high volume attacks, so a co-implementation of different saturation mitigation strategies might be a work around. The other main mitigation strategies proposed by Pascoal are derivatives of a Moving Target Defense (MTD) strategy, to randomly update the timeouts of flow rules, the IP and Mac addresses and other parameters to force the attackers to improve their methodology.

Overall, many of these strategies present the same problems, the fact that a device can be blocked instantaneously after an algorithm labels it as an attack implies that the model’s metrics for false positives and negatives will have a huge impact on the performance of the whole Intrusion Prevention System, which can lead to undesired consequences for the legitimate users. These decisions to block after a single message definitively have their use cases and are understood as a measure to keep the network’s uptime and stability on positive numbers, but if there is to achieve a balance between the two of this factors, new strategies must be employed.

The main difference of this work from the previous ones is that this project will focus mainly on slow-rate type DDoS attacks that, apart from the typical traffic mitigation threshold for both benign and malicious flow, will consider the controller performance. The mitigation strategy works on top of an existent IDS and uses a variety of techniques, not just the usual approach to identify and drop network packets.

From the previous works we can get some interesting ideas to mitigate slow DDoS attacks, such as the use of blacklists and safe lists to control access to the network resources, from Pascoal’s work we can get some interesting ideas to mitigate slow rate DDoS attacks, as we can draw a parallel between the flow rule generation to overwhelm the network to the generation of slow open connections on a server. Other useful proven techniques are Moving Target Defense.

Objectives

The general objective of this project is to implement a mitigation strategy for slow-rate DDoS attacks. This strategy will be analyzed, and its results will be reported in a technical article. The general objective was broken down into the specific objectives shown in Table I.

Tabla

Descripción generada automáticamente

TABLE I. Project Objectives

Deliverables

Problem Justification

As mentioned previously, despite the usefulness an SDN provides to big corporations and data centers, there are security vulnerabilities that must be addressed. For example, the controller is always an attack target, because the whole network is centralized [2]. Furthermore, the number of DDoS incidents has risen exponentially from year to year [10]. According to Kaspersky, the second quarter of 2020 saw an increase of 300\% the number of regular DDoS attacks compared to the previous year, and an increase of over 200\% of smart DDoS attacks [11]. These factors emphasize the need for new mitigation strategies to defeat DDoS attacks.

Gráfico, Gráfico de barras

Descripción generada automáticamente

Figure X. Kaspersky report on DDoS attacks in 2020

Mitigation strategies have not kept up with the rate of complexity of modern DDoS attacks. Most mitigation strategies are focused primarily on volumetric attacks, and there is little research on slow-rate DDoS, further extenuating the need for the development of new ways to attenuate attacks.

Architecture

The architectural design for this project will be borrowed from Pérez-Díaz et al. [8]. And it consists of the separation of the detection and mitigation modules while being technologically agnostic, so any of the two can be implemented in about any programming language, so if a technology or framework is better suited to deal with certain tasks, its implementation can go ahead without worrying too much on the integration.

The three main modules are:

1. Flow Management module, its task is to detect HTTP flows.
2. Suspicious Attackers Management module, it keeps a list of attackers.
3. Mitigation module, it installs flow rules for malicious flow mitigation.

The IDS is interfaced to allow communication with the IPS, and it classifies a given flow into labels for normal or attack traffic with the use of a machine learning model.

//Replace Figure X

Figure X shows the design for the architecture.

The steps for a flow to be mitigated are as follows:

1. The SDN installs a flow rule needed to detect and process incoming HTTP flows.
2. An incoming flow queries the flow table, and if a key is matched, the flow is then sent to the SDN controller through a secure channel.
3. The Flow management module creates and sends in JSON format, object data from the flow’s headers for identification of the ML model to use.
4. The Identification API selects a model and performs classification, sending the response back to the IPS.
5. The Suspicious Attackers module keeps data about potential malicious traffic, always looking for signals to create the flow blocking rule.
6. The mitigation flow rules are installed inside the ONOS controller.

Imagen que contiene Diagrama

Descripción generada automáticamente

Figure X. Project architecture

Proposed Solution

Many current articles state that the problem with different mitigation strategies is that the attacker still occupies a lot of resources while the detection and mitigation phases occur. This strategy seeks to prevent, or at least diminish, the impact of the attacker based on how many resources they consume.

The strategy consists of adding the detected attacker to a blacklist with two parameters: a starting rate of 30 and a starting factor of 5 points, and it will be forwarded as usual. Each time it is detected again, its factor will increase according to the difference between its current delay and the network’s timeout configured value, where an upwards increasing formula, shown on figure X, will be used to increase it during each detection, and this factor will be summed to the rate until a threshold is reached. At that point, a flow rule will be installed.

Texto

Descripción generada automáticamente

Figure X. Factor-increasing formula

The basic idea behind this is to “punish” the flows that are more demanding on the network, where a packet delay that is closer to a configured timeout will increase the individual factor faster than one that is farther from it.

Here are the steps of this strategy, closely following the diagram for the strategy in Figure X:

1. A flow is passed to the IDS and classified whether it is “normal” or an attack, and it is returned to the network as a JSON object. If it is not an attack, the algorithm ends, and the traffic is forwarded to its destination as usual.
2. [Follows from step 1] If the label is attack, and it is the first time it has ever been detected, an entry will be created for the “Attack Stats Table”, where the key is a string with the IP direction pair of source-destination.
3. Along with adding a flow key to the table, three parameters are used: a rate with 30 starting points, and a factor, starting at 5 points and a timestamp, logging the exact time in milliseconds the attack has been executed (this value is provided by the server). At this point the flow will be forwarded.
4. [Follows from step 1] If the attacker has been detected before, and there is an entry for it in the “Attack Stats Table”, the strategy will read the values associated with it from the table, read the new timestamp and calculate a delay (the time between each attack).
   1. After it is done, the formula on figure X is used to update the factor, the formula was designed with the intention to increase the factor in percentage points after each consecutive detection, table X shows how much the factor would be increasing after a given delay.
   2. A middle step of minimizing the delay between it and TIMEOUT – 1 is done to avoid potentially high resulting factor, product from dividing by an extremely small number.
   3. The factor is then summed to the rate and the values for that entry key are updated.
5. If the new rate is less than a threshold (set at 100), the attack will be forwarded.
6. If the new rate is equal to or exceeding the threshold, a flow rule blocking this attacker is installed with a duration of 7 days (although this value is flexible), and the attacker is deleted from the “Attack Stats Table” so it does not take unnecessary space, waiting for it to undergo the whole algorithm again if it ever attacks again once the flow rule expires.

Tabla

Descripción generada automáticamente

Table X. Factor increase value given any delay, set 10 as TIMEOUT example.

Imagen que contiene Escala de tiempo

Descripción generada automáticamente %3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%26lt%3Bfont%20color%3D%26quot%3B%23ffffff%26quot%3B%26gt%3B4.1%26lt%3Bbr%26gt%3B%26lt%3B%2Ffont%26gt%3B%22%20style%3D%22ellipse%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfillColor%3D%233112FF%3BstrokeColor%3D%23F0FFFF%3B%22%20vertex%3D%221%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22510%22%20y%3D%22670%22%20width%3D%2230%22%20height%3D%2230%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3C%2Froot%3E%3C%2FmxGraphModel%3E

Figure X. Flow Diagram of the mitigation strategy

|  |
| --- |
| **Algorithm** 1: Mitigation Strategy |
| **input**: A JSON file with an attacker’s info.  **output**: Attack Stats Table  **Result**: Blocking or forwarding of traffic |
| //Parameters and constants  THRESHOLD←100;  BASE\_FACTOR←5;  STARTING\_RATE←30;  TIMEOUT←10ms;  //Algorithm Begins here  **If** Flow.key in Attack\_stats\_table **then**  FlowVals ← Attack\_stats\_table [Flow.key];  Timenow ← Flow.timestamp;  lastTime ← FlowVals.lastTime;  delay ← timenow − lastTime;  delay ← min(delay, T IM EOU T−1);  rateIncrease←10s;  oldF actor ← FlowV als.f actor;  newFactor ← old\_Factor×(1 +rateIncrease);  oldRate ← FlowVals.rate;  newRate←oldRate+newF actor;  **If** newRate >= THRESHOLD **then**  add block rule to flow table;  delete Flow.key from Attack\_stats\_table;  **continue**;  **end**;  FlowV als.lastTime←timenow;  FlowVals.rate←newRate;  FlowV als.f actor←newF actor;  save FlowVals in Attack\_stats\_table [flow.key];  **else**  create Attack\_stats\_table [Flow.key];  FlowVals ← Attack\_stats\_table [Flow.key];  FlowVals.factor ← BASEFACTOR;  FlowVals.rate ← STARTING\_RATE;  FlowVals.lastTime ← Flow.timestamp;  Save FlowVals in Attack\_stats\_table [flow.key];  **end** |

Advantages

Contrary to some mitigation strategies discussed on the “State of the Art” section, this technique actually takes into consideration the false positives the IDS predicts, and makes it perform better than just a first detection, if only examining the benign vs the malicious flows, as a few detection are needed to block a device, which gives it room for error if it accidentally predicts a positive attack when there is none.

The best advantage is its behavior. The “punishment” that every flow goes through makes sense from an intuitive point of view: if you know that a device is consuming more resources than others (by sending the packets more slowly) then you know that it must be penalized more severely, by making its factor and its rate increase faster.

Disadvantages

This strategy runs into the problem discussed at the end of the “State of the Art” section, where the server suffers from the consequence of not blocking potentially malicious traffic at the first time. At first glance, the number of packets the network has to process before acting up gives us the idea that it may affect the availability of the server while the attack is being processed, which would not happen with faster and more concise strategies.

Tests

Configuration Setup

The experiments will be done on a virtual machine with an ubuntu testbed. The SDN Controller will be done using ONOS and the Mininet program will be used for the network simulation.

The system consists of 2 main custom programs developed with the maven tools to fit with the Architecture of the project:

* Flow Collector: processes the incoming requests and interfaces with the IDS to classify any given flow.
* Reactive Forwarding: here is where the mitigation strategy lives, and it begins by consulting the prediction log from the server to process the possible attacks.

The following network topology will be used:

Diagrama

Descripción generada automáticamente

Figure X. A partially connected topology

Experiments

A series of tests will be performed with the topology illustrated on Figure X. The methodology for the tests will be to let run normal traffic for several flows over a period, execute the attacks and wait for the network to mitigate them and only process normal traffic again. This is done to better visualize the network under different kinds of device traffic.

The tests consist of running slow-rate attacks using the Slow-HTTP-Test tool. Some of the hosts will act as server victims and others will act as the attackers.

The metrics to be considered for the evaluation will be:

* Number of malicious packets dropped and not dropped.
* Number of benign packets dropped and not dropped.
* Time until the network returns to the normal state (after the attackers have been blocked).

The roles of the hosts will be as follows:

Normal traffic generation:

Receivers: PC1, PC2, PC3, PC4

Senders: PC9, PC10, PC11, PC12

Attack traffic:

Victims: PC5, PC6, PC7, PC8

Attackers: PC13, PC14, PC15, PC16

The attacks to consider are as follows (r stands for rate of packets per second; c stands for number of connections open):

* Slow – read: r = 300 and c = 2000, r = 200 and c = 3000
* Slow – header: r = 300 and c = 2000, r = 200 and c = 3000
* Slow – body: r = 300 and c = 2000, r = 200 and c = 3000

Evaluation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Attack | Total Malicious pkts | Malicious pkts dropped (%) | Total Legit pkts | Legit pkts processed | Time until return to normal state |
| Slow read: r=300 and c=2000 | 35851 total  583 processed | 98.37% | 659 total  0 dropped | 100% | 10s |
| Slow – read: r = 200 and c = 3000 | 33791 total  445 processed | 98.68% | 621 total  79  dropped | 87.27% | 12s |
| Slow – body: r = 200 and c = 3000 | 24358 total  171 processed | 99.29% | 141  40 |  | 123s |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Conclusions and Future Work

Conclusions

Future Work

Any mitigation strategy that works with the same architecture in this work will only be as good as the Intrusion Detection System as the network can only drop packets based on the response from the IDS server. That is why for this project, including the whole network programs, should be included when improving its performance.

Some things that can be done for this project include:

* Try new machine learning models and deep learning architectures.
* For the specific strategy on this project, look for ways to further “punish” attacking flows, involving other network resources like the number of connections opened per device.
* Evaluate the performance of the strategy on offline projects or on datasets.
* Optimize the starting parameters from the algorithm.

References

[1] F. S. Dantas Silva, E. Silva, E. P. Neto, M. Lemos, A. J. Venancio Neto, and F. Esposito, “A Taxonomy of DDoS Attack Mitigation Approaches Featured by SDN Technologies in IoT Scenarios,” Sensors, vol. 20, no. 11, p. 3078, May 2020.

[2] Business, C., 2020. 12 DDoS Statistics That Should Concern Business Leaders. [online] Coxblue.com. Available at: <https://www.coxblue.com/12-ddos-statistics-that-should-concern-business-leaders/> [Accessed 27 February 2021].

[3] Cisco Visual Networking Index: Forecast and Trends (2017–2022). Available online: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html (accessed on 15 December 2019).

[4] M. V. O. Assis, L. F. Carvalho, J. Lloret, and M. L. Proenca jr., “A GRU deep learning system against attacks in software-defined networks”, Journal of Network and Computer Applications, vol. 177, p. 102942, mar. 2021, doi: 10.1016/j.jnca.2020.102942.

[5] N. N. Tuan, P. H. Hung, N. D. Nghia, N. V. Tho, T. V. Phan, and N. H. Thanh, “A DDoS Attack Mitigation Scheme in ISP Networks Using Machine Learning Based on SDN,” Electronics, vol. 9, no. 3, p. 413, Feb. 2020, doi: 1ñ0.3390/electronics9030413.

[6] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proenca jr., “Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment”, IEEE Access, vol. 8, pp. 83765–83781, 2020, doi: 10.1109/access.2020.2992044.

[7] Hong, K., Kim, Y., Choi, H., and Park, J. (2018). SDN-Assisted Slow HTTP DDoS Attack Defense Method. IEEE Communications Letters, 22(4), 688–691. doi:10.1109/lcomm.2017.2766636

[8] Perez-Diaz, J. A., Valdovinos, I. A., Choo, K.-K. R., and Zhu, D. (2020). A Flexible SDN-based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks using Machine Learning. IEEE Access, 1–1. doi:10.1109/access.2020.3019330

[9] Pascoal, T. A., Fonseca, I. E., and Nigam, V. (2020). Slow Denial-of-Service Attacks on Software Defined Networks. Computer Networks, 107223. doi:10.1016/j.comnet.2020.107223

[10] REAL security. 2019. Evolution of DDoS in the last decade - REAL security. [online] Available at: <https://www.real-sec.com/2019/08/evolution-of-ddos-in-the-last-decade/> [Accessed 19 February 2021].

[11] KUPREEV, O., BADOVSKAYA, E. and GUTNIKOV, A., 2021. DDoS attacks in Q2 2020. [online] Securelist. Available at: <https://securelist.com/ddos-attacks-in-q2-2020/98077/> [Accessed 20 February 2021].

[12] Lukaseder, T., Ghosh, S., & Kargl, F. 2018. Mitigation of flooding and slow DDoS attacks in a software-defined network. [online] arXiv preprint arXiv:1808.05357. Available at: <https://arxiv.org/abs/1808.05357/> [Accessed 15 May 2021]